PCT National Publication Gazette

National Patent Publication No.	2000-507015
Date of National Publication:	June 6, 2000
International Class(es):	G 06 F 9/45

(37 pages in all)

Title of the Invention:	Real Time Program Language Accelerator
Patent Appln. No.	10-517009
Filing Date:	November 13, 1997
Date of Filing Translation:	May 12, 1999
International Filing No.	PCT/US97/20980
International Publication No.	WO98/21655
International Publication Date:	May 22, 1998
Priority Claimed:	Country: U.S.A. Filing Date: November 13, 1996 Serial No. 60/030,688
Inventor(s):	RAZ, Yair
Applicant(s):	PARAN, Arik (transliterated, therefore the spelling might be incorrect)

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 11/00 — A1

(11) International Publication Number: **WO 98/21655**

(43) International Publication Date: 22 May 1998 (22.05.98)

(21) International Application Number: PCT/US97/20980

(22) International Filing Date: 13 November 1997 (13.11.97)

(30) Priority Data:
60/030,688 13 November 1996 (13.11.96) US

(71) Applicant *(for all designated States except US)*: PARAN, Arik [US/US]; 866 Helena Drive, Sunnyvale, CA 94087 (US).

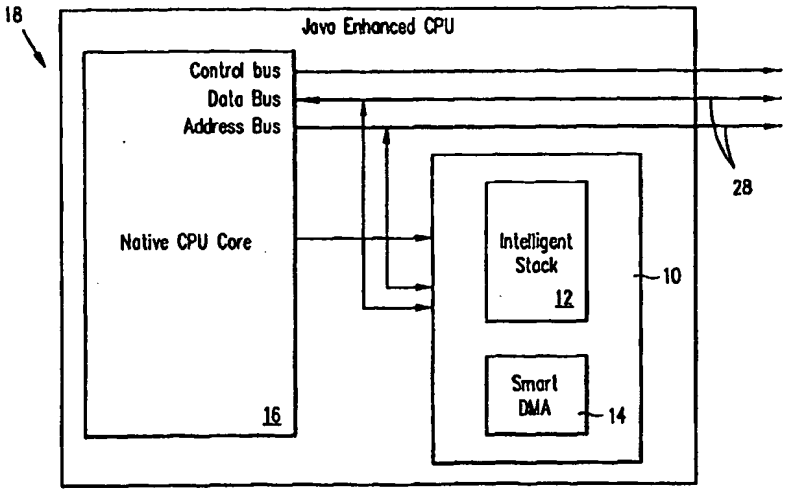
(71)(72) Applicant and Inventor: RAZ, Yair [IL/US]; 1575 Lewiston Drive, Sunnyvale, CA 94087 (US).

(74) Agents: HAMRICK, Claude, A., S. et al.; Oppenheimer Wolff & Donnelly LLP, Suite 600, Ten Almaden Boulevard, San Jose, CA 95113 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, US, UZ, VN, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: REAL TIME PROGRAM LANGUAGE ACCELERATOR



(57) Abstract

A program language accelerator core (10) operating in conjunction with a program language accelerator software (34) in a computer (20) for optimizing the operation of Java™ program code in a native CPU (16) processor. An intelligent stack (12) has a plurality of memory areas (52, 54, 56 and 58) in a memory map (50) mapped such that particular operations are performed depending upon which of the memory areas (52, 54, 56 or 58) the native CPU (16) addresses. When the program language accelerator software (34) translates the Java™ code into the native code of the native CPU (16), optimization is provided by directing read and write operations to the appropriate area in the memory map (50) such that it is not necessary to include a separate instruction regarding how the data being written is to be handled.

(19)日本国特許庁（ＪＰ）　(12)公表特許公報（Ａ）　(11)特許出願公表番号
特表2000－507015
（P2000－507015A）
(43)公表日　平成12年６月６日(2000.6.6)

(51)Int.Cl.'	識別記号	Ｆ Ｉ	テーマコード'(参考)
Ｇ０６Ｆ　9/45		Ｇ０６Ｆ　9/44	３２２Ｆ

審査請求　有　　予備審査請求　有　（全 37 頁）

(21)出願番号　特願平10－517009
(86)（22)出願日　平成９年11月13日(1997.11.13)
(85)翻訳文提出日　平成11年５月12日(1999.5.12)
(86)国際出願番号　ＰＣＴ／ＵＳ９７／２０９８０
(87)国際公開番号　ＷＯ９８／２１６５５
(87)国際公開日　平成10年５月22日(1998.5.22)
(31)優先権主張番号　６０／０３０，６８８
(32)優先日　平成８年11月13日(1996.11.13)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　ラツ，ヤイール
アメリカ合衆国　カリフォルニア　94087，サニーベイル，ルイストン　ドライブ　1575
(71)出願人　バラン，アリック
アメリカ合衆国　カリフォルニア　94087，サニーベイル，ヘレナ　ドライブ　866
(72)発明者　ラツ，ヤイール
アメリカ合衆国　カリフォルニア　94087，サニーベイル，ルイストン　ドライブ　1575
(74)代理人　弁理士　山本　秀策

最終頁に続く

(54)【発明の名称】　リアルタイムプログラム言語アクセラレータ

(57)【要約】
プログラム言語アクセラレータコア（１０）は、固有ＣＰＵ（１６）プロセッサでJava™プログラムコードの動作を最適化するために、コンピュータ（２０）においてプログラム言語アクセラレータソフトウェア（３４）との組み合わせで動作する。インテリジェントスタック（１２）は、マッピングされるメモリマップ（５０）において複数のメモリ領域（５２、５４、５６および５８）を有することによって、固有ＣＰＵ（１６）がどちらのメモリ領域（５２、５４、５６および５８）をアドレスするかに依存して特定の動作が実行される。プログラム言語アクセラレータソフトウェア（３４）がJava™コードを固有ＣＰＵ（１６）の固有コードに翻訳する際、書き込まれたデータがどのように処理されるかに関連する個別の命令を含む必要がないメモリマップ（５０）の適切な領域に読み出しおよび書き込み動作を向ける事によって最適化が提供される。

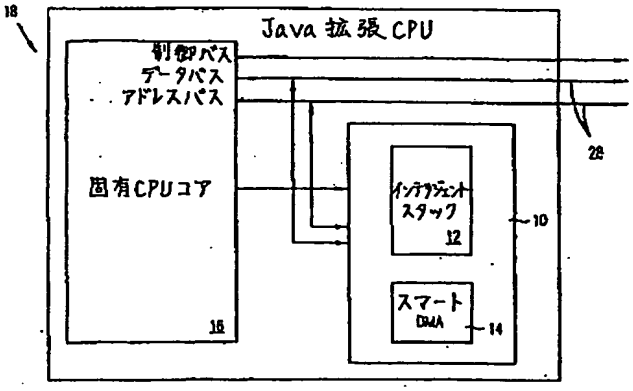


FIG. 1

【特許請求の範囲】

１．スタック指向の解釈上の言語命令の高速化処理を提供するコンピュータシステムであって、

固有言語命令を実行する固有処理装置と、

該解釈上の言語命令を該固有言語命令に翻訳し、該固有言語命令のうちの選択された命令であって関連するアドレス値を発生するトランスレータであって、各該選択された命令および該対応するアドレス値の各々が対応するコア動作に関連するトランスレータと、

該処理装置と通信するために結合された処理アクセラレータコアであって、

該アドレス値の対応するものによって示される複数のメモリアドレスロケーションを有するスタックキャッシュメモリであって、データを格納するための該ロケーションが該コア動作の対応するものと関連する、スタックキャッシュメモリと、

該アドレス値（および該固有処理装置によって提供される読み出し／書き込み制御信号）に反応し、該キャッシュメモリ装置を往復する該選択された固有言語命令と関連する該データの書き込みおよび読み出しを制御するように作用するコア制御論理と、

該キャッシュメモリ装置の該アドレスロケーションの対応するものからの該データを受け取り、該データを用いてコア動作を実行するために作用する演算論理回路であって、該コア動作の各々が該対応するアドレスロケーションによって特定化される、演算論理ユニットと、を含む処理アクセラレータコアと、

を含む、コンピュータシステム。

２．前記トランスレータが前記固有処理装置によって実行される翻訳命令によって実現される、請求項１に記載のコンピュータシステム。

３．前記コア制御論理が

スタック方法論に従って前記キャッシュメモリ装置を往復する該データの前記書き込みおよび呼び出しを制御するスタックコントローラと、

前記対応するコア動作を決定する目的で前記アドレス値の各々を復号化するデ

コーダと、

を含む、請求項１に記載のコンピュータシステム。

４．主要メモリユニットを更に含み、前記コア制御論理が、前記キャッシュメモリ装置がスタックオーバーフロー状態に近づいた時に該主要メモリユニットに該キャッシュメモリ装置からデータをシフトするように作用し、該キャッシュメモリ装置がスタックアンダーフロー状態に近づいた時に、該主要メモリユニットから該キャッシュメモリ装置にデータをロードするように作用するダイレクトメモリアクセスコントローラを更に含む、請求項１に記載のコンピュータシステム。

５．前記ダイレクトメモリアクセスコントローラが、

Variable Load Missの現象において前記主要メモリユニットから前記キャッシュメモリ装置にローカル変数をロードし、

Variable Store missの現象においてローカル変数を主要メモリユニットに格納し、

コンテクスト切換動作について準備するために該キャッシュメモリ装置から該主要メモリユニットにデータを転送し、

必要に応じて、該キャッシュメモリ装置に入出力するデータを移動することによって高速化スレッドコンテクスト切換を実行する、

ように更に作用する、請求項４に記載のコンピュータシステム。

６．前記アドレス値の各々が、

前記キャッシュメモリ装置の複数のメモリスペース領域から選択する複数の選択領域ビットであって、該領域の各々が前記アドレスロケーションの特定のセットを含む複数の選択領域ビットと、

該領域の各々の該アドレスロケーションの各々の特定のものを選択し、実行す前記動作を決定する、複数の選択ビットと、

を含む、請求項１に記載のコンピュータシステム。

７．前記解釈上の言語がJava™である、請求項１に記載のコンピュータシステム。

８．前記コア動作が算術演算を含む、請求項１に記載のコンピュータシステム。

９．前記コア動作がブール（Boolean）論理動作を含む、請求項１に記載のコンピュータシステム。

１０．スタック指向の解釈上の言語命令の高速化処理を提供する方法であって、固有言語命令を実行する固有処理装置と、該処理装置と通信するために結合される処理アクセラレータコアと、該コアが、複数のメモリアドレスロケーションと、前記キャッシュメモリ装置を往復するデータの書き込みおよび読み出しを制御するように作用する制御論理と、該キャッシュ装置の該アドレスロケーションの対応するものからデータを受け取る演算論理回路とを有し、該データを用いてコア動作を実行するように作用するスタックキャッシュメモリ装置を含む、処理アクセラレータコアと、を含むコンピュータシステムにおいて、

解釈上の言語命令を受け取る工程と、

該解釈上の言語命令を該固有言語命令に翻訳する工程と、

該固有言語命令の選択されたものと関連するアドレス値を発生する工程であって、該選択された命令の各々および該対応するアドレス値が対応するコア動作に関連する工程と、

該キャッシュメモリ装置のロケーションを往復するデータを書き込みおよび読み出しする工程であって、該ロケーションが該アドレス値の対応するもので示され、該データが該コア動作の対応するものと関連する工程と、

該キャッシュ装置の該アドレスロケーションの対応するものから該演算論理回路に該データを提供し、該データを用いて該コア動作を実行するために該演算論理回路を用いる工程であって、該コア動作の各々が該対応するアドレスロケーションによって特定される工程と、

を含む方法。

１１．前記翻訳する工程が、前記固有処理装置を用いる実行命令を含む、請求項１０に記載のコンピュータシステム。

１２．前記コア論理が、

スタック方法論に従って前記キャッシュメモリ装置を往復するデータの前記書き込みおよび読み出しを制御するスタックコントローラと、

前記対応するコア動作を決定する目的で前記アドレス値の各々を復号化するデコーダと、

を含む、請求項１０に記載のコンピュータシステム。

１３．前記キャッシュメモリ装置がスタックオーバーフロー状態に近づいた時に該キャッシュメモリ装置から前記コンピュータシステムの主要メモリユニットにデータをシフトする工程と、

該キャッシュメモリ装置がスタックアンダーフロー状態に近づいた時に該主要メモリユニットから該キャッシュメモリ装置にデータをロードする工程と、

を含むダイレクトメモリアクセス制御工程を更に含む、請求項１０に記載のコンピュータシステム。

１４．Variable Load Missの現象において前記主要メモリユニットから前記キャッシュメモリ装置にローカル変数をロードする工程と、

Variable Store missの現象においてローカル変数を該主要メモリユニットに格納する工程と、

コンテクスト切換動作について準備するために該キャッシュメモリ装置から該主要メモリユニットにデータを転送する工程と、

必要に応じて、該キャッシュメモリ装置を出入りするデータを移動することによって高速化スレッドコンテクスト切換を実行する工程と、

を含む、ダイレクトメモリアクセス制御工程を更に含む、請求項１３に記載のコンピュータシステム。

１５．前記アドレス値の各々が、

前記キャッシュメモリ装置の複数のメモリスペース領域から選択する複数の選択領域ビットであって、該領域の各々が前記アドレスロケーションの特定のセットを含む複数の選択領域ビットと、

該領域の各々の該アドレスロケーションの各々の特定のものを選択し、実行される前記コア動作を決定する複数の選択ビットと、

を含む、請求項１０に記載のコンピュータシステム。

１６．前記解釈上の言語がJava™である、請求項１０に記載のコンピュータシス

テム。

１７．前記コア動作が算術演算を含む、請求項１０に記載のコンピュータシステム。

１８．前記コア動作がブール論理動作を含む、請求項１０に記載のコンピュータシステム。

【発明の詳細な説明】

リアルタイムプログラム言語アクセラレータ

技術分野

本発明は、コンピュータ処理の分野に関し、より詳細には、独自のハードウェアおよびソフトウェアの組合せを用いるコンピュータコードのリアルタイム解釈および動作に関する。本発明のリアルタイムプログラム言語アクセラレータの主な現在の使用は、Java™コードにおいての実行であり、多様な異なるプロセッサ上でこのようなコードを高い実行速度で実行できることが望ましい。

技術背景

当該技術において、より高いレベルのコンピュータ言語をほぼリアルタイムで機械読取可能コードに変換するインタープリタを提供することが公知である。しかし、このようなインタープリタは、プログラムのローディングおよび実行中の少なくともいくつかの点で実行されるプログラムの動作を遅らせる必要がある。特に、Java™プログラム言語に関しては、「Java™ Virtual Machine」を含むJava™プログラムを実行するいくつかの利用可能な解決策は、それらのタスクを達成するために実行が必要なソフトウェアベースのプログラムである。

任意の利用可能なタイプのプロセッサ上でJava™コードを実行する方法および／または手段を有することが有益である。また、それが実行するより実際に速く実行するまたは少なくともJava™コードを解釈および実行するプロセス全体を減速しない、Java™コードを処理するいくつかの方法および／または手段を有することが有益である。しかし、本発明者の知る限りでは、従来技術においてこのような目的を達成するシステムは存在しない。確かに、Java™ Virtual machineなどのソフトウェア解決策は、コードの動作を最適化するように試みるが、限界がコードが実行される特定のプロセッサおよびシステムの特徴であった。

発明の開示

よって、本発明の目的は、事実上いかなるタイプのプロセッサにおいても非固有プログラム言語を容易に実行する方法および手段を提供する事である。

本発明の別の目的は、Java™プログラムコードの動作を高速化する方法および

手段を提供することである。

本発明のさらに別の目的は、ハードウェア上で容易に実現される、Java™コードを翻訳および実行する方法および手段を提供することである。

本発明の別の目的は、コンピュータシステムが非Java™コードを実行する能力を低減しない、Java™コードを翻訳および実行する方法および手段を提供することである。

簡単に、本発明の好適な実施形態は、ＣＰＵチップの一部として個別のチップ上、または個別のボード上でさえ実施され得る集積回路「プログラム言語アクセラレータコア」である。プログラム言語アクセラレータコアは、ハードウェアスタックメモリの専用であり、Java™コードの実行のために存在またはエミュレートされる必要のあるスタックメモリを直接提供する。ダイレクトメモリアクセス（「ＤＭＡ」）コントローラも提供され、オーバーフローおよびアンダーフロー条件（または他の特異的な条件）が生じる際にスタックメモリに対してデータをシフトする。本発明のソフトウェア部分は、Java™コードを機械の固有言語コードに翻訳し、また、特定の機能を実行するように前もって決められたメモリアドレスを書き込むなど、必要に応じて本発明に独自の機能を実行するためにコードを最適化する。この方法において、それ以外ではいくつかのクロックサイクルを要し得る動作は、単一の（または少なくともより少ない）クロックサイクルで実行され得る。

本発明の利点は、Java™コードを実行する速度が非常に増加する事である。

本発明の更なる利点は、Java™コードが、本質的に任意のタイプのプロセッサ上で容易に実行され得ることである。

本発明の更に別の利点は、安価なコンピュータであってもJava™コードの実行のために最適化され得るように、本発明が容易および安価に実現され得ることである。

本発明の更に別の利点は、今後入手可能になり得るような新しいタイプのプロセッサに適応することが困難または高価ではないことである。

本発明のこれらの、および他の利点は、本発明を実行するのに現在知られてい

る最良の方法、および本明細書中に記載およびいくつかの図面に示される好適な実施形態の産業的利用性を考慮して当業者に明らかになる。

図面の簡単な説明

図１は、本発明によるプログラム言語アクセラレータコアを上に有する集積回路のブロック図である。

図２は、本発明により向上されたＣＰＵがコンピュータシステムにどのように統合されるかの例を示すブロック図である。

図３は、図１のインテリジェントスタックのより詳細なブロック図である。

図４は、図１および図３に示されるインテリジェントスタックのメモリマップである。

図５は、本発明のプロセスの１つの局面のフローチャートである。

図６は、本発明による値プッシュ動作のフローチャートである。

図７は、本発明による値ポップ動作のフローチャートである。

図８は、本発明による定数プッシュ動作のフローチャートである。

図９は、本発明による算術演算のフローチャートである。

図１０は、本発明によるローカル変数ロード動作のフローチャートである。

図１１は、本発明によるローカル変数格納動作のフローチャートである。

本発明を実行する最良の方法

本発明を実行するために現在知られている最良の方法は、リアルタイムアクセラレータコアである。本発明のリアルタイムアクセラレータコアを図１にブロック図として示し、その中で全体的な参照符号１０で示す。

リアルタイムアクセラレータコア１０は、インテリジェントスタック１２およびスマートＤＭＡコントローラ１４を有する。リアルタイムアクセラレータコア１０は、固有ＣＰＵが追加のソフトウェアベースの翻訳または解釈なしにJava™プログラムを（標準Java™インタープリタまたはＪＩＴコンパイラと比較して）より速い実行モードで実行し得るように標的固有ＣＰＵ１６によってJava™オブジェクトコードのリアルタイム解釈および実行することを可能にする。図１の例において、プログラム言語アクセラレータコア１０およびＣＰＵは単一のＣ

ＰＵチップ１８上で実施されているが、本明細書中で上述したようにこれは本発明の必要な局面ではないことに留意されたい。

図２は、内部にプログラム言語アクセラレータ１０を有するＣＰＵチップ１８で動作するように構成された典型的なコンピュータの図表の例である。コンピュータチップ内のＣＰＵチップ１８の接続は、同種の従来のコンピュータ（図示せず）と大差はない。データバス２２、アドレスバス２４および制御バス２６は、適切なデータ経路２８にそれぞれ設けられ、ＣＰＵチップ１８、（ＲＡＭ）メモリ３０、およびＩ／Ｏセクション３２と通信する。

データバス２２は、固有ＣＰＵデータバスである。データバス２２の幅は、固有ＣＰＵ１６の自然データバス幅（８、１６、または３２ビット）であるべきである。プログラム言語アクセラレータコア１０におけるインタフェース論理は、タスクに関する任意のデータ幅を処理する。アドレスバス２４は、固有ＣＰＵアドレスバスである。アドレスバス２８の幅は、固有ＣＰＵ１６の自然アドレスバス幅であるべきである。制御バス２６は、クロック、リセット、読取、書き込み、割り込みライン、バス要求など、いかなるＣＰＵでも見出され得るいくつかの制御信号、を輸送する。

図２の例は、本発明のプログラム言語アクセラレータコア１０が用いられ得るコンテクストを示すためのみに提供されており、本発明の局面を開示するように意図されない。動作中、メモリ３０は、固有ＣＰＵ１６（図１）のための要求された初期化ソフトウェア、（用いられる場合は）オペレーティングシステム、Ｉ／Ｏ装置ドライバ、または制御コード、ならびにいかに更なる詳細で議論される機能を有するプログラム言語アクセラレータソフトウェア３４を含む。更に、メモリ３０は必要とされる任意のコンパイルされたJava™コード（例えば、特殊カスタムクラスライブラリ）を含み得る。

エンドユーザアプリケーションに依存して、本発明のプログラム言語アクセラレータコア１０およびプログラム言語アクセラレータソフトウェア３４を装備しているコンピュータ２０は、不揮発性メモリに既に存在する任意のJava™コードを実行し得るか、またはそれを適用可能なＩ／Ｏ装置（通信ポート、ディスクな

ど）を介してロードし得る。

プログラム言語アクセラレータコア１０は、Java™ Virtual Machineがスタックベースの機械エミュレーションであり、（図１の固有ＣＰＵコア１６等の）現代のＣＰＵ上のスタック動作のほとんどが１クロックサイクル以上かかって完了する事実を利用する。インテリジェントスタック１２は、本質的にJava™ Virtual Machineスタックのための大きな仮想キャッシュである。ＤＭＡコントローラ１４（図１）は、以下に詳細に議論されるように実際のキャッシュの制限されたサイズによってスタックオーバーフローまたはアンダーフローが生じる場合にキャッシュを一貫して保つ。

図３は、図１に関連して本明細書中に前述されたインテリジェントスタック１２のより詳細なブロック図である。図３を見ると、インテリジェントスタック１２が簡単な内部演算論理回路（「ＡＬＵ」）を有することが分かる。この回路は、Java™ Virtual Machineの実現において当業者によく知られたiaddまたはiincなどのスタックの上部(tops)をアップデートする動作と共にJava™ Virtual Machlneの仕様書によって規定されるような定数のプッシュを可能にする。キャッシュ３８は、インテリジェントスタック内の実際のキャッシュメモリである。スタックコントローラ４０は、従来のスタック制御装置において見出されるようなスタックポインタ４２およびスタックポインタネクスト４４を有する。従来のデコーダ４６がアドレスおよび命令を復号化するために提供され、出力ＭＵＸ４８が動作に適切なインテリジェントスタック１２内のソースから出力を提供する。

インテリジェントスタック１２は、スタックとして用いられるように構成された６４×３２ビットレジスタファイルである。インテリジェントスタック１２は、固有ＣＰＵ１６メモリアドレススペースにマッピングされたメモリであり、固有ＣＰＵ１６の速いスタック動作（プッシュ、ポップ）および、レジスタの任意の１つへのランダムアクセスを可能にする。インテリジェントスタック１２は、自動的なスタックオーバーフロー／アンダーフロー検出のための回路を提供する。スマートＤＭＡコントローラ１４は、異なるタスクの間をシフトする必要がある

時などにメモリ３０に対してまたはメモリ３０からインテリジェントスタック１２の全体のコンテントを読み出したり書き込んだりするように構成されている。スマートＤＭＡコントローラ１４は、また、インテリジェントスタック１２からメモリ３０への任意の過剰なデータを一時的に格納することによって潜在的なスタックオーバーフローまたはアンダーフローを訂正する。スマートＤＭＡは、スタックがオーバーフローに近づいた際（プッシュ動作の際）にメインメモリ３０にワードブロックをダンプするか、または、スタックがアンダーフローに近づいた際（ポップ動作の際）にメインメモリからワードブロックをロードする。スマートＤＭＡコントローラは、（変数ロードミスの際に）メインメモリ３０からのローカル変数をロードするか、（変数格納ミスの際に）メインメモリ３０にローカル変数を格納するか、コンテクスト切換のために調整されたメインメモリ３０にキャッシュ３８の全体をダンプするか、または、コンテクスト切換を実行するためにメモリ３０からキャッシュ３８の全体をロードする事ができる。また、スマートＤＭＡは、必要に応じてキャッシュ３８に対してデータを移動することによってスレッドコンテクスト切換を高速化するために任意に用いられ得る。

本発明のプログラム言語アクセラレータ１０は、スマートＤＭＡコントローラ１４の追加によってプログラム言語アクセラレータ１０の有用性および機能性を顕著に増加させるが、スマートＤＭＡコントローラ１４なしで動作させ得ることに留意されたい。

Java™動作がスタック機械アーキテクチャに基づくことから、プログラム言語アクセラレータソフトウェア３４はJava™コードから固有ＣＰＵコードへの非常に単純な翻訳のみを実行することを要求される。翻訳は、好ましくはJava™コードがロードされる際に、支持言語アクセラレータソフトウェア３４を用いて行われる。本発明による最適化は、以下により詳細に議論されるように、特定のメモリアドレスに指示される特定の動作を提供する本発明の方法を含む方法による翻訳の間に提供される。固有ＣＰＵ１６は（インテリジェントスタック１２によって処理される）スタック管理に対処する必要がないので、Java™プログラムの非常に高速な実行が可能である。

本発明によれば、データはデータバス２２から来て（従来のスタック動作と同

様に）スタックの上部に流れるか、または、スタックの最高部における値を含む算術演算を有し、次いでスタックの上部の上に置換またはプッシュする。本発明に独自であるのは、動作の全ては用いられるアドレスによって決定されるということである。インテリジェントスタック１２は、キャッシュ３８のアドレススペースの４倍を占有する。すなわち、インテリジェントスタック１２における実際のメモリロケーションの全てを完全にアドレスするのに必要な数よりも４倍のアドレスがＣＰＵチップ１８のメモリにマッピングされる。本発明の現在知られる最良の実施形態１０のキャッシュのサイズは、６４ワードであり、この実施形態においてインテリジェントキャッシュ１２は２５６個のロケーションを占有する。

図４は、インテリジェントスタック１２によって占有されたメモリアドレスの単純なメモリマップ５０である。図４を見ると、メモリマップ５０は、第１の領域５２、第２の領域５４、第３の領域５６、および第４の領域５８を有することが分かる。これらのうち、第１の領域５２のみが物理キャッシュ３８に直接関連する。メモリマップ５０をアドレスするための完全なアドレス６０は、領域５２、５４、５６、または５８のうちどの領域がアドレスされるか選択するための２つの選択領域ビット６２および、それぞれの領域内の特殊データロケーションをアドレスするための８ビット選択バイト６４を有する。４つの領域５２の各々は、その中にあるアドレスに対して読み出しまたは書き込みをすることによって実行される異なる動作を可能にするため、プログラム言語アクセラレータ１０の上部２つのアドレスビット（選択領域ビット６２）はどの領域５２、５４、５６、または５８がアドレスされたかを決定し、装置の主要動作モードを決定する。

下記の表１は、固有ＣＰＵ１６がメモリマップ５０の４つの領域５２、５４、５６および５８の各々をアドレスする際に実行される動作の全体的な説明を提供する。

表１

上位アドレスビット（６２）	プログラム言語アクセラレータコア主要動作領域
00	ハードウェアスタック（キャッシュ）

01	データに対する直接アクセス
10	構成および制御レジスタに対するアクセス
11	読み出し：スタックからの値をポップする。書き込み：（領域内のアドレスに依存して）スタックへの定数または値をプッシュする
	読み出し：定義されない書き込み。要求された動作結果とTOSを置換する。動作はTOSおよび書き込まれたデータ上で実行される。動作は領域内のアドレスに依存する

簡単に、メモリマップ５０の第１の領域５２に書き込まれた値は、装置のベースアドレスにおいて固有ＣＰＵ１６によって直接書き込みまたは読み出され得る。第２の領域５４は、構成および制御レジスタに使用される。第３の領域５６は、プッシュおよびポップ動作のために用いられる。第４の領域５８は、スタックの上部で値を置換する算術演算に使用される。デコーダ４６は、入来アドレス要求を考察し、それにより動作を自動的に決定する。アドレスと動作との間の相関関係は、プログラム言語アクセラレータソフトウェア３４による翻訳中に事前に確立されている。データバスおよびタイミングは、アドレス、読み出しおよび書き込み信号がクロック上で有効であり、読み出しデータが後に続くクロックによって予期される包括的な単一サイクルバスである。

図５は、本発明の方法によるコンピュータ２０の一般的な動作を示すフローチャート６８である。コンピュータ２０が再起動されるときはいつでも、固有ＣＰＵ１６によって特定の初期化が行われる：ハードウェア装置の設定、（使用される場合は）オペレーティングシステムの初期化、任意の要求されたスタートアッププログラムの実行、等。これらのプログラムは、固有ＣＰＵ１６言語（例えば、命令セット）に書き込まれ、固有ＣＰＵ１６によって直接実行される。これらは、

Java™および／またはプログラム言語アクセラレータコア１０と全く関係無しに行われる。また、内部レジスタ動作７０の設定において、固有ＣＰＵは、プログラム言語アクセラレータコア１０をアドレスするために取っておかれたメモリレジスタを含むメモリレジスタを初期化する。これはユーザがJava™ Execution Environmentを呼び出して起動するときの動作の初期段階である。Java™ Virtual Machineの初期化中に、Java™ Virtual Machineはプログラム言語アクセラレータコア１０およびその支持ソフトウェア（プログラム言語アクセラレータソフトウェア３４）の存在を検出し、それら両方を初期化する。この段階で行われる様々な初期化のうち、最も重要なもののいくつかは（本明細書中にメモリマップ５０の第２の領域５４を関連して説明された）プログラム言語アクセラレータコア１０の様々な構成レジスタ設定である。

次いで、任意の必要な固有プログラムがコンピュータ２０によって実行され得る。これは、プログラム言語アクセラレータコア１０による任意の動作を必要とせずに固有ＣＰＵ１６によって直接行われる。（これらの動作は、図５のフローチャート６８に図示されておらず、フローチャート６８を正しいコンテクストに置くためだけにここで述べられる。）

（メモリ内に既に存在するか、またはディスクからロードされるあるいは通信ラインから受信される必要がある）Java™のクラスファイルが実行される必要があるとき、固有ＣＰＵ１６はＪＶＭクラスローダおよびプログラム言語アクセラレータソフトウェア３４を用いて要求されたファイルをロード、準備、翻訳の実行、および実行をし始める。このプロセスは、後に続くセクションで詳細に説明されるいくつかのステップから形成される。クラスファイルロード動作７２において、Java™クラスファイルがロードされる。この部分は、「Java™ Virtual Machine Specification」に記載されるように標準Java™ Virtual Machineコードによって全部が実行される。連結および照合動作７４において、連結、照合、準備および決定は、Java™ Virtual Machineリンカーによって従来通り実行される。この部分も、標準Java™ Virtual Machineコードによって全部が実行される。

翻訳動作７６に続いて、固有ＣＰＵ１６がクラスファイルの実現コード（例え

ば、クラスを実現するJava™ Virtual Machineバイトコード）を見つけ、固有の命令に翻訳し、メモリ３０における実行領域にロードする。この部分は、プログラム言語アクセラレータソフトウェア３４によって全部が行われる。翻訳段階の目的は、（前の段階でロードされ、連結された）Java™バイトコードを固有ＣＰＵの命令ストリームに変換し、適切にプログラム言語アクセラレータコア１０を動作することである。本明細書中に記載のように、特別のメモリロケーションから／に対して読み出しまたは書き込むことは、プログラム言語アクセラレータコア１０の動作の全てを呼び出すため、翻訳された固有コードは、プログラム言語アクセラレータコア１０の様々なメモリ領域から／に対する読み出しおよび書き込み命令を主に含む。バイトコードを固有コードに特異的に翻訳することは、プログラム言語アクセラレータコア１０に付属する固有ＣＰＵに依存する。翻訳されたコードは、メモリ３０に格納される。一旦クラスファイルが完全に翻訳されると、オリジナルバイトコードイメージが破棄され得、翻訳された固有コードのみが用いられる。

一旦ローディングプロセスが完了すると、固有ＣＰＵ１６は翻訳された固有コードのエントリポイントへの分岐（ジャンプ）を実行し、クラス初期化コードを実行し始める。これは、本開示を通して論じられるように、ハードウェアスタックおよび論理専用のプログラム言語アクセラレータコア１０を利用する命令シーケンスを有する固有プログラムである。これは、図５のフローチャート６８における実行固有コード動作７８によって示される。

実行固有コード動作７８内で生じる本発明の方法の動作の更なる詳細は、以下の追加のフローチャートに関連して論じられる。これらの動作およびそれぞれのフローチャートは以下の通りである：固有ＣＰＵ１６がインテリジェントスタック１２のメモリマップ５０の第２の領域５４をアドレスする際に生じる動作に関連する値プッシュ動作８０（図６）および値ポップ動作８２（図７）。定数プッシュ動作８４（図８）はまた、固有ＣＰＵがインテリジェントスタック１２のメモリマップ５０の第２の領域５４の選択されたロケーションをアドレスする際に生じる動作に関連する。（Java™に精通している者は、等価の「定数ポップ」動作が必要ないことを認識するであろう。）

算術演算８４フローチャート（図９）は、メモリマップ５０の第３の領域５４の選択された領域がアドレスされた際にプログラム言語アクセラレータコア１０で生じる動作を説明する。ローカル変数ロード動作８８（図１０）およびローカル変数格納動作９０（図１１）は、これらのそれぞれの機能を説明し、スマートＤＭＡコントローラ１４の動作について更なる詳細を提供する。

本明細書の前に紹介されたメモリマップ５０の４つの領域５２、５４、５６、および５８の異なる機能の論議に再び戻ると、本発明の現在知られている最良の実施形態において、プログラム言語アクセラレータコア１０はそのハードウェアインテリジェントスタック１２において６４ワードを含むことが思い出される。このアドレススペースは、その中のアドレスに対して読み出しまたは書き込む事によって異なる動作が実行されることを可能にする４つの主要領域５２、５４、５６、および５８のそれぞれに分割される。メモリマップ５０の第１の領域５２に関して、この領域は、この領域がメモリマップの唯一の領域であると仮定した場合、すなわち、メモリアマップ５０のアドレスとキャッシュ３８との間に１対１の相互関係が存在すると仮定した場合に、当業者がメモリマップ５０が振る舞うように期待する機能を果たす。メモリマップ５０の第１の領域５２は、それらがランダムアクセスメモリであるかのように、ハードウェアスタックレジスタ（６４－３２－ビットレジスタ）のいずれにも読み出しおよび書き込みアクセスを可能にするために提供される。

メモリマップ５０の第２の領域５４に関して、この領域におけるレジスタは、プログラム言語アクセラレータコア１０の動作を制御する。これらのレジスタは読み出しおよび書き込みアクセスを有する。また、この領域は４つの特別な書き込み専用ｅロケーションを含む。書き込み専用ロケーションの２つは、変数動作に用いられ、後の２つはコンテクスト切換に用いられる。これらのレジスタのアドレス指定は、固有ＣＰＵ１６の低いアドレスビットを介して行われる。以下の表２はメモリマップ５０の第２の領域５４内の適用レジスタのリストである。

表２

アドレス	レジスタ	レジスタまたは機能

0	SP	スタックポインタ
1	BOSP	スタックポインタの底部
2	WLIMIT	スタックが書き込まれ得る主要メモリの上限
3	RLIMIT	スタックが読み出され得るメモリにおける下限。
4	VREG	変数アドレスレジスタ。このロケーションは、試験目的のために書き込みまたは読み出しされ得る。最後にアクセスされた変数のアドレスに自動的に書き込まれる。
5	SWAPINSIZE	スワップインサイズ。このレジスタは、スワップインＤＭＡサイクルが起動された時にいくつのワードが読み出されるのかを制御する。
8	VLOAD	変数ロード。このレジスタは、書き込む専用アクセスを有する。この特別なロケーションが書き込まれた時、データバスはスタック上部にプッシュされるべきローカル変数の絶対アドレスを有する。変数がキャッシュに

		ない場合、ＤＭＡサイクルが起動される。変数がキャッシュにある場合、スタック上部のコンテンツを転送するために、１つの追加サイクルが取得される。
9	VSTOTE	変数格納。このレジスタ書き込み専用アクセスを有する。この特別なロケーションが書き込まれた時、データバスはスタック上部からの値を得るべきローカル変数の絶対アドレスを有する。ＤＭＡサイクルは、変数がキャッシュにない場合に起動される。変数がキャッシュにある場合、スタック上部のコンテンツを転送するために、１つの追加サイクルが取得される。
14	SWAPIN	スワップイン。このレジスタは書き込み専用アクセスを有する。この特別なロケーションが書き込まれた時、データバス上の値は無視され、ブロックモードＤＭＡサイクル

		が主要メモリからキャッシュを満たすために起動される。読み出されるワードの数は、SWAPINSIZEレジスタの値に依存し、スレッドの特異的な状態に調整され得る。
15	SWAPOUT	スワップアウト。このレジスタは書き込み専用アクセスを有する。この特別なロケーションが書き込まれた場合、データバス上の値は無視され、ブロックモードＤＭＡサイクルが主要メモリにキャッシュを流すために起動される。読み出されるワードの数は、SPおよびBOSPの値に依存する。

メモリマップの第３の領域５６に関して、この領域は、値または定数をスタックにプッシュするために用いられ、またスタックから値をポップするために用いられる。実行される動作は、プロセッサの低いアドレスビット（選択バイト６２）によって決定される。すなわち、アドレスが実行される動作を決定する。これらの機能は、下記の表３に列挙されている。当業者は、いくつかの場合において、固有ＣＰＵ１６によってプログラム言語アクセラレータコア１０に提供されるデータの値は、動作が一定値で実行されているため不適切であることを認識するであろう。

表３

アドレス	動作	動作の説明
0	PUSHPOP	データパス上の値をスタックにプッシュする（書き込み動作）、またはTOSからの値をデータパスにポップする（読み出し動作）
2	ICONSTm1	書き込み専用ロケーション。一定の整数（-1）をプッシュする。
3	ICONST0	書き込み専用ロケーション。一定の整数０をプッシュする。
4	ICONST1	書き込み専用ロケーション。一定の整数１をプッシュする。
5	ICONST2	書き込み専用ロケーション。一定の整数２をプッシュする。
6	ICONST3	書き込み専用ロケーション。一定の整数３をプッシュする。
7	ICONST4	書き込み専用ロケーション。一定の整数４をプッシュする。
8	ICONST5	書き込み専用ロケーション。一定の整数５をプッ

		シュする。
11	FCONST0	書き込み専用ロケーション。一定のフロート(float)0.0をプッシュする。
12	FCONST1	書き込み専用ロケーション。一定のフロート1.0をプッシュする。
13	FCONST2	書き込み専用ロケーション。一定のフロート2.0をプッシュする。

メモリマップ５０の第４の領域５８に関して、この領域は、Ｔｏｐ－Ｏｆ－Ｓｔａｃｋの値で算術演算を開始するために用いられる。スタック上部の値はデータバス上の値とＴｏｐ－Ｏｆ－Ｓｔａｃｋの現在の値との間の算術演算の結果によって置換される。実行される算術演算は、プロセッサのローアドレスビッツ（選択バイト６２）によって決定される。これらの動作の全ては、整数値（３２ビット整数）上で実行される。これらの機能は、以下の表４に列挙される。

表４

アドレス	動作	算術動作の説明
16	IADD	書き込み専用ロケーション。スタック上部の値をデータバス上の値に加え、スタック上部の値を置換する。
17	ISUB	書き込み専用ロケーション。スタック上部の値からデータバス上の値を減

		算し、スタック上部の値を置換する。
18	INEG	書き込み専用ロケーション。０からスタック上部の値を減算し、スタック上部の値を置換する。データバス上の値を無視する。
19	IOR	書き込み専用ロケーション。データバス上の値を有するスタック上部の値上でビットに関するORを実行し、スタック上部の値を置換する。
20	IAND	書き込み専用ロケーション。データバス上の値を有するスタック上部の値上でビットに関するANDを実行し、スタック上部の値を置換する。
21	IEXOR	書き込み専用ロケーション。データバス上の値に対しスタック上部の値にビットに関する排他的ORを実行し、スタック上部の値を置換する。
22	IINCR	書き込み専用ロケーション。スタック上部の値に

		対して１を加え、スタック上部の値を置換する。データバス上の値を無視する。

図６に示す値プッシュ動作８０を参照に、プログラム言語アクセラレータソフトウェア３４が値プッシュ動作に関してJava™コードを翻訳し、最適化した時、およびそのような動作が実行される時期である時、動作８０ａにおいて値がロケーション「プッシュポップ」（表３を参照）に書き込まれる。動作８０ｂにおいて、アドレスは（図３のデコーダ４６によって）複号化され、値がそのように指示される。次いで、動作８０Ｃにおいて、データバス（図２）から書き込まれた値は、そのＴＯＳレジスタによって指定されたロケーションでハードウェアスタック（図３のキャッシュ３８）に書き込まれ、ＴＯＳが増加される。ハードウェアスタック（キャッシュ３８）が決定動作８０ｄで決定されたようにオーバーフローに近づいたとき、動作８０ｅにおいて、スマートＤＭＡコントローラ１４が開始され、キャッシュ３８のコンテントの一部をメモリ３０（図２）に保存する。

図７に示す値ポップ動作８２を参照に、プログラム言語アクセラレータソフトウェア３４が値ポップ動作に関してJava™コードを翻訳し、最適化した時、およびそのような動作が実行される時期である時、動作８２ａにおいて読み出し命令が適切なロケーションに向けられる。動作８２ｂにおいて、アドレスが（図３のデコーダ４６において）複号化される。次いで、動作８２ｃにおいて、キャッシュ３８（図３）およびＴＯＳポインタから読み出された（ポップされた）値は減少される。動作８２ｄにおいて、値がデータバス（図２）に送信される。これにより、決定動作８２ｅによって決定されるようにハードウェアスタック（キャッシュ３８）がアンダーフロー状態になる場合（すなわち、キャッシュ３８が未使用の所定レベルに達成した場合）、動作８２ｆにおいて、スマートＤＭＡコントローラ１４が開始され、メモリ３０（図２）からキャッシュ３８のコンテントの一部を復元する。

⑦ 図８の一定のプッシュ動作８４を参照に、プログラム言語アクセラレータソフトウェア３４が一定のプッシュ動作に関してJava™コードを翻訳し、最適化した時、およびそのような動作が実行される時期である時、動作８４ａにおいて書き込み命令が適切なロケーションに向けられる。動作８４ｂにおいて、アドレスが（図３のデコーダ４６によって）複号化される。次いで、動作８４ｃにおいて、選択された特定のアドレスによって検出された一定値は、そのＴＯＳレジスタによって示されたロケーションにおいてハードウェアスタック（図３のキャッシュ３８）に書き込まれ（プッシュされ）、ＴＯＳが増加される。ハードウェアスタック（キャッシュ３８）が決定動作８４ｄで決定されたようにオーバーフローに近づいたとき、動作８４ｅにおいて、スマートＤＭＡコントローラ１４が開始され、キャッシュ３８のコンテントの一部をメモリ３０（図２）に保存する。

⑧ 図９の算術演算８６を参照に、プログラム言語アクセラレータソフトウェア３４が特定の算術動作に関してJava™コードを翻訳し、最適化した時、およびそのような動作が実行される時期である時、動作８６ａにおいて書き込み命令が適切なロケーションによって向けられる。その適切なロケーションが一体なんであるのかは、本明細書において表４で前に列挙したように、実行される特定の算術演算によって検出される。動作８６ｂにおいて、アドレスが（図３のデコーダ４６によって）複号化され、対応するデータがデータバス２２（図２）からフェッチされる。次いで、動作８６ｃにおいて、選択されたアドレスに対応する算術演算は、データバス（図２）から書き込まれた値を用いてスタック上部における値上で実行され、そのＴＯＳレジスタによって示されたロケーションにおいてハードウェアスタック（図３のキャッシュ３８）に結果が書き込まれる。

⑨ 変数格納動作９０（図１１）において、プログラム言語アクセラレータソフトウェア３４が変数格納動作に関してJava™コードを翻訳し、最適化した時、およびそのような動作が実行される時期である時、動作９０ａにおいて、インテリジェントスタック１２の格納動作に対応する専用メモリロケーションに書き込みが向けられる。データバス２２に書き込まれる値は、格納されるべき必要なローカル変数の絶対メモリアドレスであるべきである。動作９０ｂにおいて、アドレス

は（図３のデコーダ４６によって）複号化される。次いで、（決定動作９０ｃによって決定されるように）必要な変数がスタックにある場合、変数値がＴＯＳから読み出され、必要なアドレスに格納され、スタック上部が減少される。動作

８８ｃで決定されるように必要な変数がスタックにない場合、動作９０ｅにおいてスマートＤＭＡコントローラ１４が開始され、スタック上部からの変数値を格納し、ＴＯＳを減少する。決定動作９０ｆによって決定されるようにハードウェアスタック（キャッシュ３８）がアンダーフローに近づいた場合、スマートＤＭＡコントローラ１４が開始され、主要メモリ３０からスタックコンテントを復元する。

変数ロード動作８８において（図１０）、プログラム言語アクセラレータソフトウェア３４が変数ロード動作に関してJava™コードを翻訳し、最適化した時、およびそのような動作が実行される時期である時、動作８８ａにおいて、変数格納動作専用のインテリジェントスタック１２の専用メモリロケーションに書き込みが向けられる。データバス２２に書き込まれた値は、格納されるために必要なローカル変数の絶対メモリアドレスであるべきである。動作８８ｂにおいて、アドレスは（図３のデコーダ４６によって）複号化される。次いで、（決定動作８８ｃによって決定されるように）必要な変数がスタックにある場合、動作８８ｄにおいて変数値はキャッシュ３８から読み出され、スタック上部に置かれ、ＴＯＳが増加される。決定動作８８ｃで決定されるように必要な変数がスタックにない場合、動作８８ｅにおいて、スマートＤＭＡコントローラ１４が開始され、変数値がスタック上部にロードされ、ＴＯＳを増加する。ハードウェアスタック（キャッシュ３８）が決定動作８８ｆに決定されるようにオーバーフローに近づいた場合、スマートＤＭＡコントローラ１４が開始され、ハードウェアスタック（キャッシュ３８）から主要メモリ３０にスタックコンテントを転送する。

本発明の価値または範囲を変更することなく本発明に多様な改変が成され得る。例えば、本発明のプログラム言語アクセラレータコア１０および関連のプログラム言語アクセラレータソフトウェア３４は、本明細書中にJava™プログラム言語と使用するために最適化されるように説明されたが、関連する原理は、特に、

このような言語が主にスタックベースのシステムと使用されるために開発され得る場合は、他のプログラム言語と使用することも等しく適用可能である。

上述したように、他のありそうな改変は、それが現存システムまたは現存システム設計により容易に加えられるようにＣＰＵチップ１８から(form)物理的に異なる装置としてプログラム言語アクセラレータを実現することである。

0 産業的応用性

本発明のプログラム言語アクセラレータコア１０および関連するプログラム言語アクセラレータソフトウェア３４は、さもなければJava™以外の言語で書き込まれるプログラムを実行するために最適化されるプロセッサとの組み合わせで、Java™コードのリアルタイム実行のために広く使用されるように意図される。上述を考慮して理解されるように、プログラム言語アクセラレータコア１０は、実行のためにJava™コードを固有ＣＰＵ１６の固有コードに解釈する時、プログラム原語アクセラレータソフトウェアが、動作がインテリジェントスタック１２によって自動的に達成されるように特定の仮想メモリアドレスに向けられる特定の動作を引き起こす、全く新しい概念を用いる。すなわち、インテリジェントスタック１２は、実行されるべき動作がどれであるかを知り、データがメモリマップに書き込まれるアドレスのみに基づいてそれを実行する。これは、この特徴が呼び出された場合に、動作当たり１からいくつかのクロックサイクルを節約する。よって、Java™コードの実行の速度は、コンピュータ２０が仮想機械などをバックグラウンドで実行する負担無しに、大いに向上される。

本発明のプログラム言語アクセラレータコア１０が容易に製造され、システムおよび装置の現存する設計に統合され得、本明細書に記載の利点が提供されるので、本発明がこの産業に容易に受け入れられることが期待される。これらおよび他の理由から、本発明の利用性および産業的応用性は、範囲が重要であり期間が永続的である。

上記の全ては、本発明の利用可能な実施形態の例のほんの一部である。当業者は、本発明の精神および範囲を逸脱する事なく多くの他の改変および変更が成され得ることが容易に認めるであろう。従って、上述の開示は、制限することを意

図せず、添付の請求の範囲は本発明の範囲の全体を包含するように解釈されるべきである。

【図１】

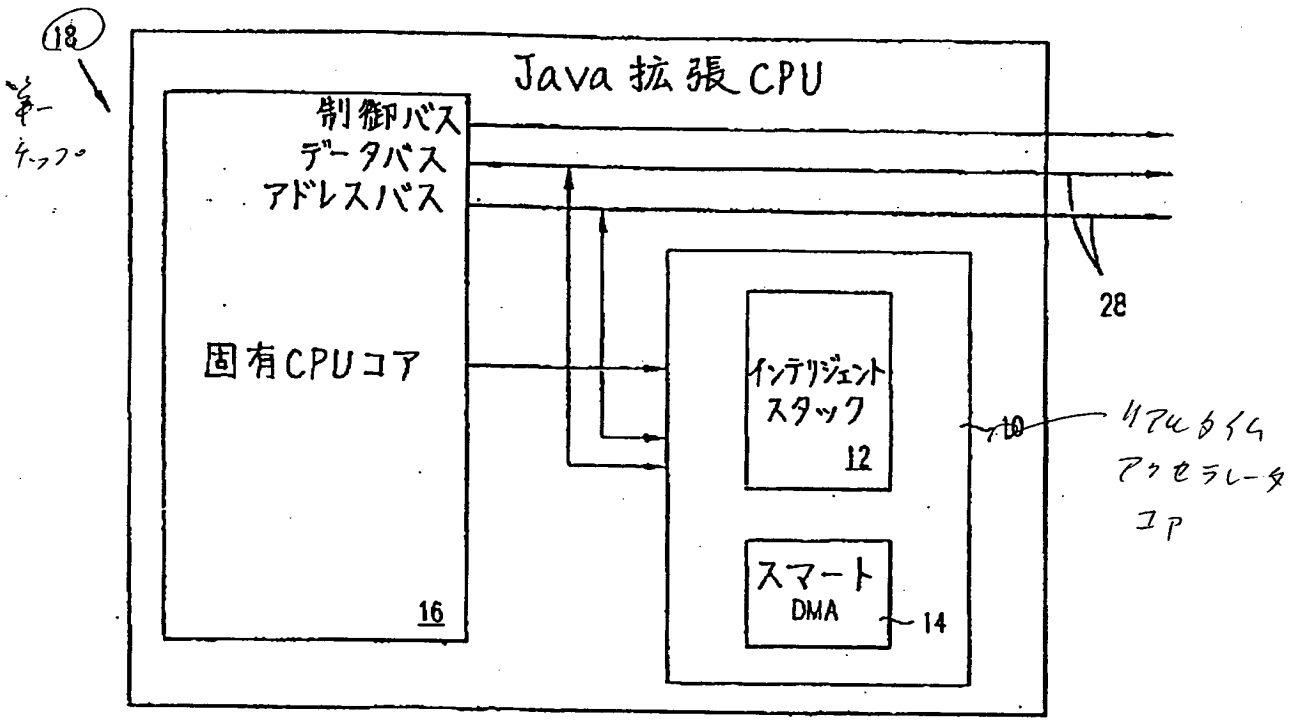


FIG. 1

【図２】

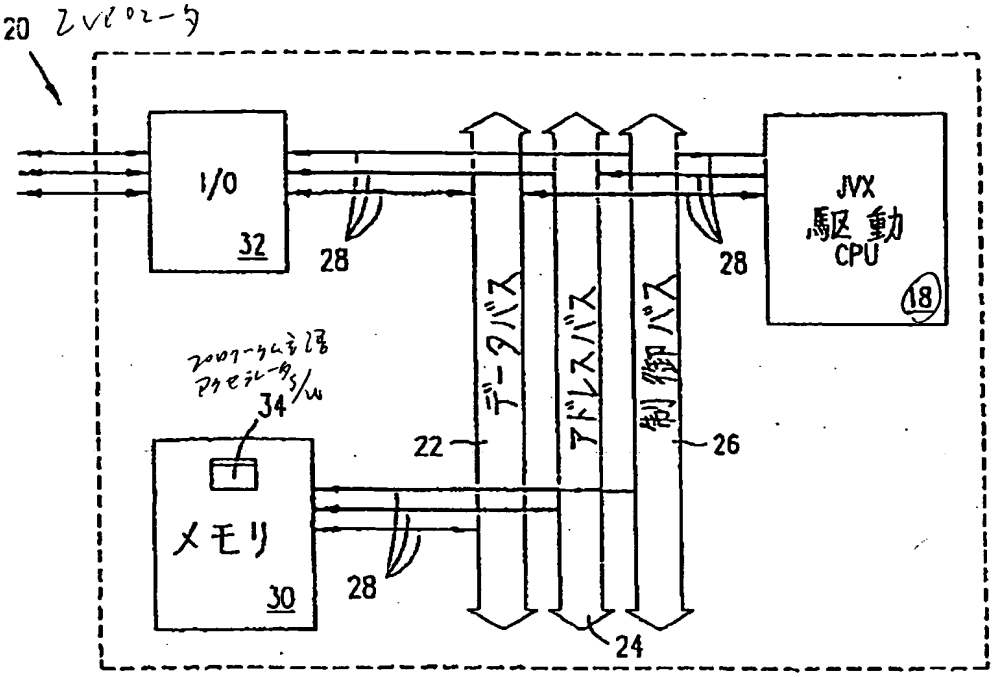


FIG. 2

【図３】

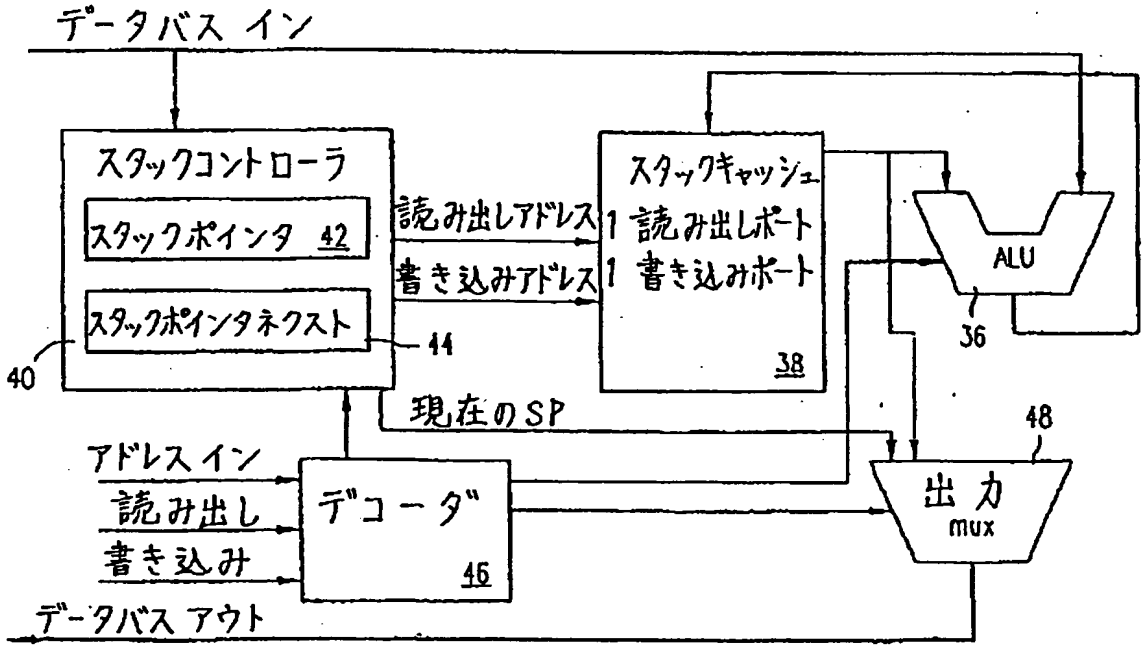


FIG. 3

【図４】

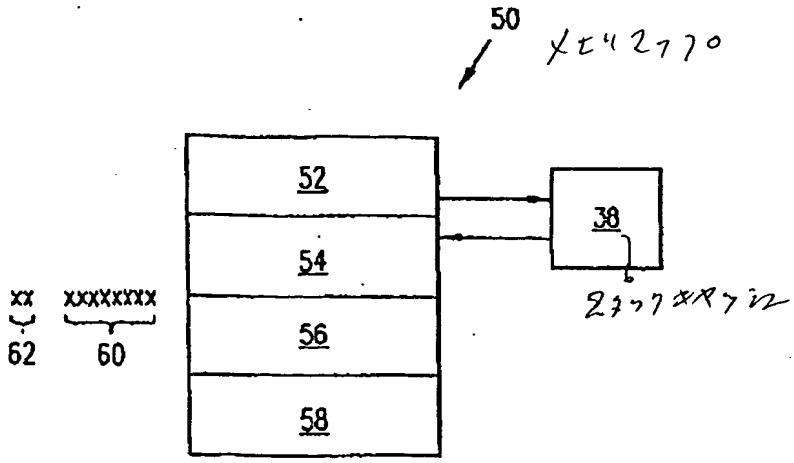


FIG. 4

【図５】

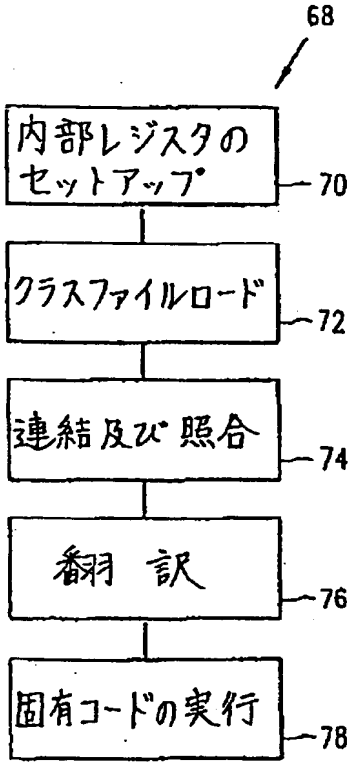


FIG. 5

【図６】

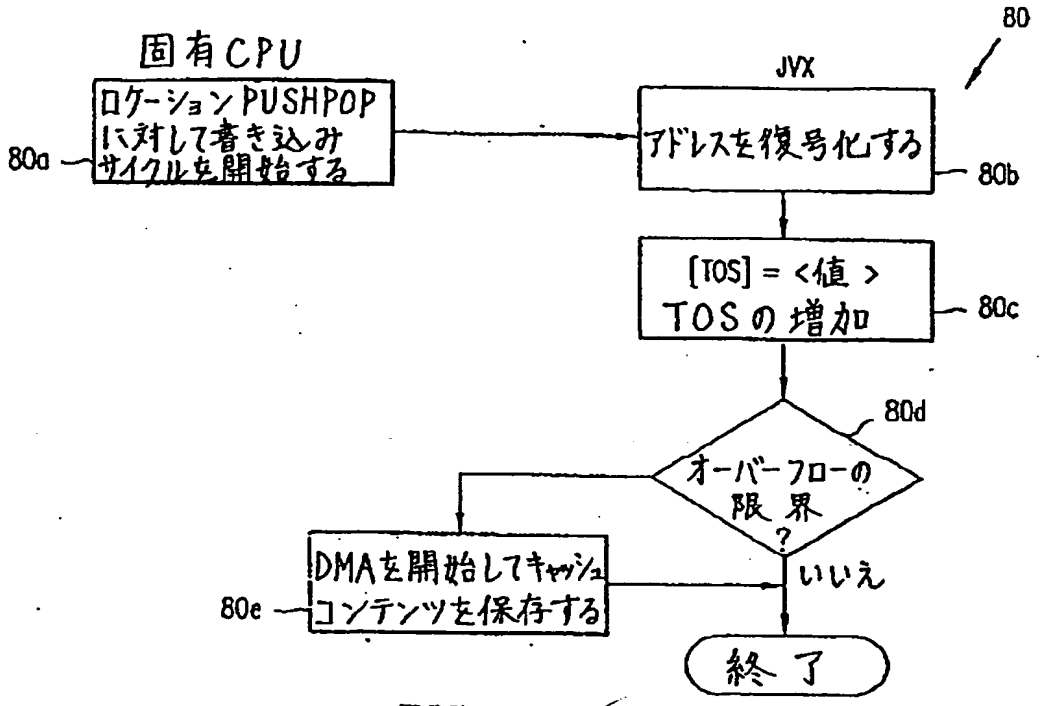


FIG. 6

【図７】

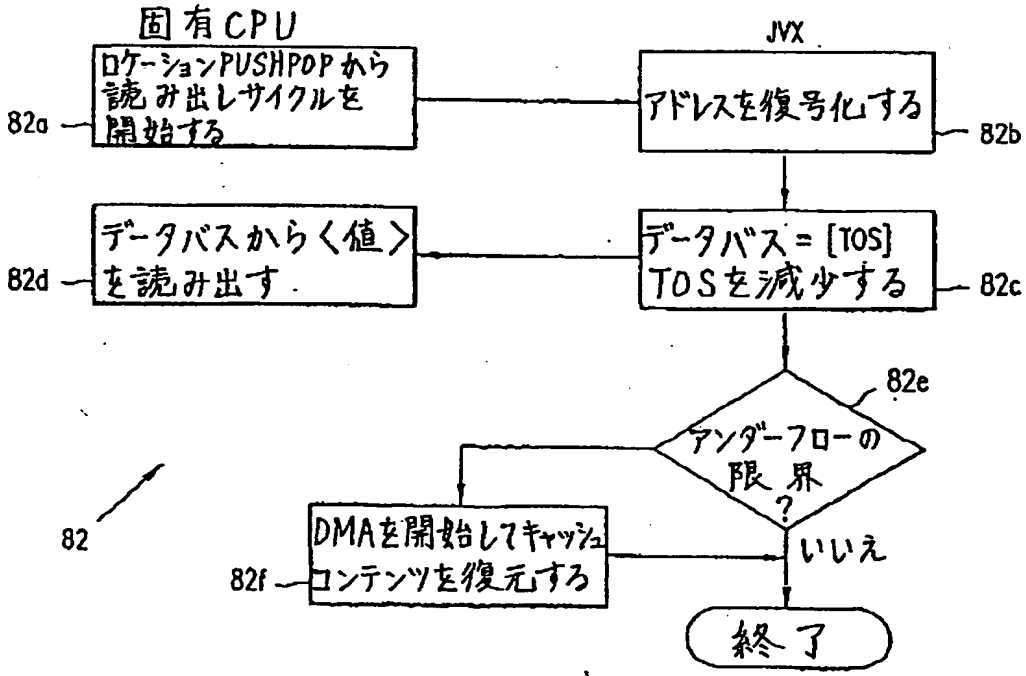


FIG. 7

【図８】

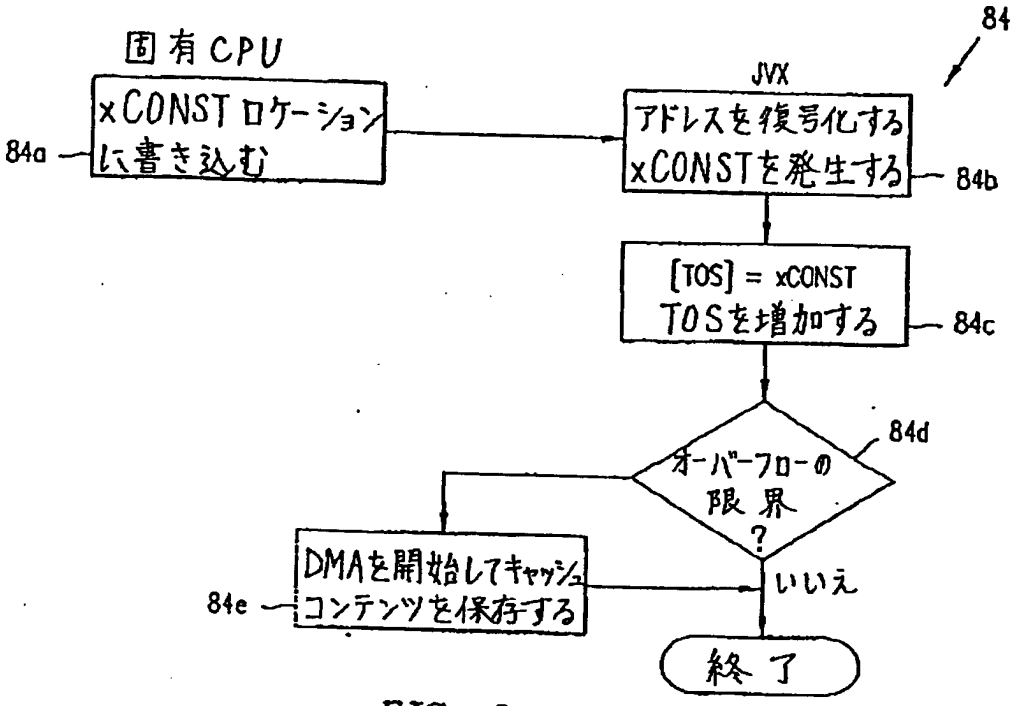


FIG. 8

【図９】

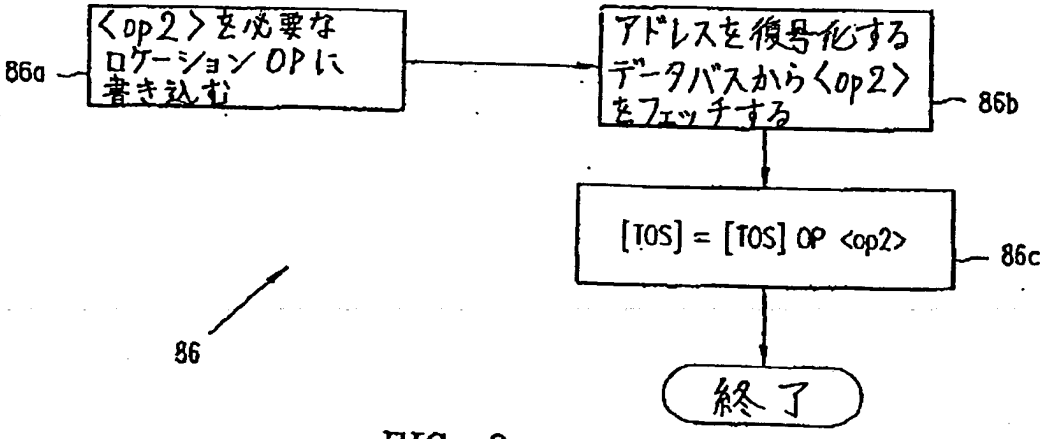


FIG. 9

【図１０】

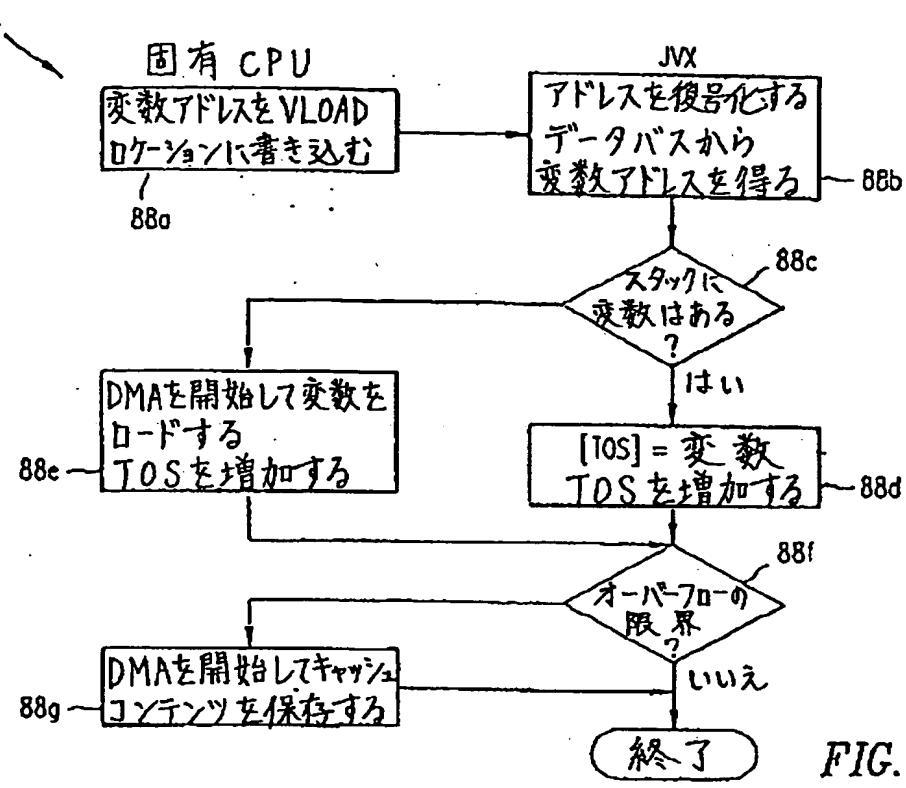

88
固有 CPU
変数アドレスをVLOAD
ロケーションに書き込む
88a
JVX
アドレスを復号化する
データバスから
変数アドレスを得る
88b
スタックに
変数はある
?
88c
はい
[TOS] = 変数
TOSを増加する
88d
DMAを開始して変数を
ロードする
TOSを増加する
88e
オーバーフローの
限界
?
88f
いいえ
DMAを開始してキャッシュ
コンテンツを保存する
88g
終了

FIG. 10

【図１１】

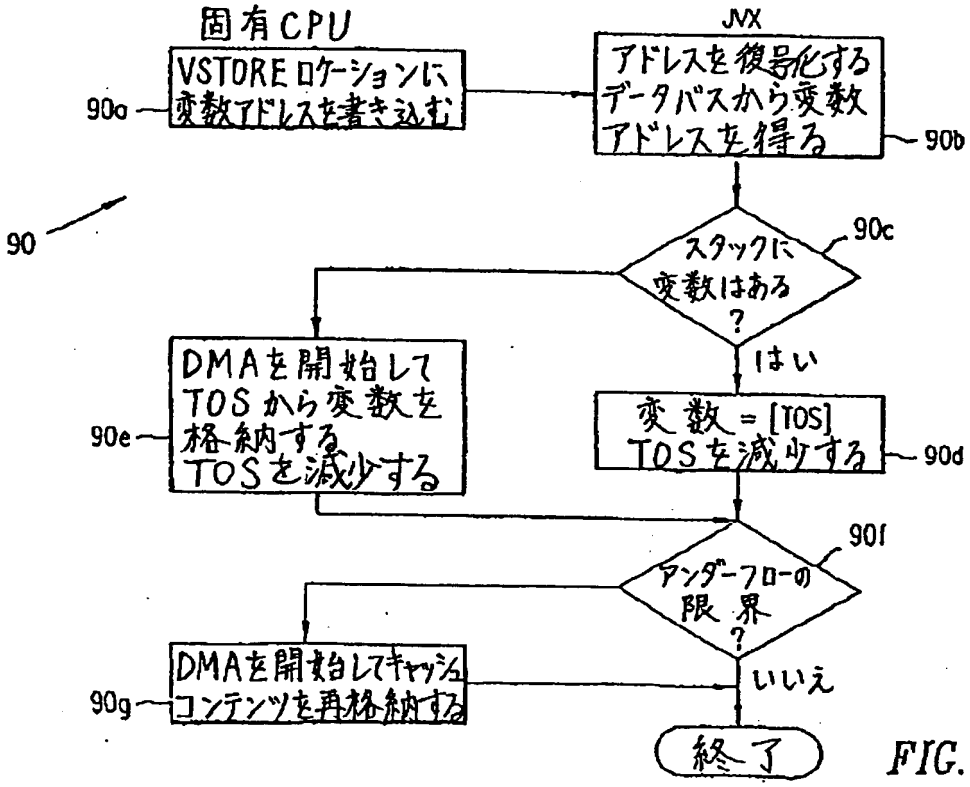

固有CPU
JVX
VSTOREロケーションに変数アドレスを書き込む
90a
アドレスを復号化するデータバスから変数アドレスを得る
90b
90
スタックに変数はある？
90c
はい
DMAを開始してTOSから変数を格納するTOSを減少する
90e
変数＝[TOS]TOSを減少する
90d
アンダーフローの限界？
90f
DMAを開始してキャッシュコンテンツを再格納する
90g
いいえ
終了

FIG. 11

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US97/20980

A. CLASSIFICATION OF SUBJECT MATTER
IPC(6) :G06F 11/00
US CL :395/705, 706, 709, 406, 381, 800.01
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 395/705, 706, 709, 406, 381, 800.01

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
APS, COMPUTER SELECT, IEEE

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CASE. B. Implementing the Java Virtual Machine; Java's Complex Instruction Set Can Be Built in Software or Hardware Microprocessor Report March 1996. Vol 10, No. 4. pages 12-18, especially pages 12 and 13.	1-14
Y	WAYNER. P. Sun Gambles on Java Chips Byte November 1996. Vol 21. No. 11. pages 79-85, especially page 82.	1-14
Y	US 4,205,370 A (HIRTLE) 27 May 1980, see entire document.	1-14
Y	US 4,674,089 A (PORET et al) 16 June 1987, see entire document.	1-14
Y	US 5,442,777 A (NAKAJIMA et al) 15 August 1995, see entire document.	1-14

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search	Date of mailing of the international search report
11 FEBRUARY 1998	14 APR 1998
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer MICHAEL RICHEY Telephone No. (703) 305-9669

Form PCT/ISA/210 (second sheet)(July 1992)*

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US97/20980

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,126,541 A (SHINAGAWA) 30 June 1992, see entire document.	1-14

Form PCT/ISA/210 (continuation of second sheet)(July 1992)*

フロントページの続き

(81)指定国 EP(AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OA(BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG), AP(GH, KE, LS, MW, SD, SZ, UG, ZW), UA(AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, US, UZ, VN